**Lab Session 05**

**Objectives**

**Arrays & Loop**

1. Indexed operands
2. Pointers
3. JMP instruction
4. LOOP instruction



1. **Indexed Operator**

An *indexed operand* adds a constant to a register to generate an effective address. Any of the 32-bit general-purpose registers may be used as index registers.

1. **Example in 16-bit mode**

**SYNTAX:**

**constant [ reg16 ] ; reg16 can either be SI, DI, BX or BP**

**[ constant + reg16 ]**

**EXAMPLE:**

.data

arrayB db 20, 40, 60, 80

arrayW dw 100, 150, 250, 300

.code

mov si, 1 ; SI = 0001

mov al, arrayB[si] ; AL = 40

mov al, [arrayB + 3] ; AL = 80

mov si, 2 ; SI = 2

mov cx, arrayW[si] ; CX = 150

mov cx, [arrayW + 4] ; CX = 250

1. **Example in 32-bit mode**

**SYNTAX:**

**constant [ reg32 ] ;reg32 can be any of the 32-bit general registers**

**[ constant + reg32 ]**

**EXAMPLE:**

.data

arrayB BYTE 20, 40, 60, 80

.code

mov esi, 0

mov al, arrayB[esi] ;AL=20

inc esi

mov al, arrayB[esi] ;AL=40

mov esi, 3

mov al, [arrayB + esi] ;AL=80

1. **Example using scale factors**

**SYNTAX:**

**constant [ reg32 \* TYPE constant]**

**EXAMPLE:**

.data

arrayW WORD 1000, 2000, 3000, 4000

.code

main PROC

mov ax, arrayW

mov esi, 1

mov ax, arrayW[esi \* TYPE arrayW]

mov esi, 2

mov ax, arrayW[esi \* TYPE arrayW]

mov esi, 3

mov ax, arrayW[esi \* TYPE arrayW]

call DumpRegs

1. **Pointers**

A variable containing the address of another variable is called a *pointer*.

**SYNTAX:**

**constant1 TYPE OFFSET constant2**

**EXAMPLE:**

.data

arrayW WORD 1000, 2000, 3000, 4000

ptrW DWORD OFFSET arrayW

.code

main PROC

mov eax, [ptrW]

1. **JMP instruction**

The JMP instruction causes an unconditional transfer to a destination, identified by a code label.

**SYNTAX:**

**JMP destination**

**EXAMPLE:**

.code

top:

; any statements

jmp top

1. **LOOP instruction**

**SYNTAX:**

**LOOP destination**

**EXAMPLE # 01:**

.data

intArray WORD 100, 200, 300, 400, 500

.code

main PROC

mov esi, 0

mov ax, 0

mov ecx, LENGTHOF intArray

L1:

mov ax, intArray [esi]

add esi, TYPE intArray

loop L1

**EXAMPLE # 02:**

mov eax, 0

mov ebx, 0

mov ecx, 5

L1:

inc eax

mov edx, ecx

mov ecx, 10

L2:

inc ebx

loop L2

mov ecx, edx

loop L1